leopard

# Leopard 极速开发手册

联系：Chen_9g (80588183@qq.com)

源码： https://github.com/chg122345/leopard.git

Chen_9g

2018-4-20

# 1. 声明

(1).手写框架不是单纯为了写框架，而是为了更好的理解，学习框架的使用及其原理。

(2).该项目只是实现了相应的功能，并没有去做相应的代码优化，毕竟水平太有限，时间也仅用 14 天完成，后续会陆续优化修改。

(3).该项目仅采用简单工厂，单列设计模式。（这方面有待去学习）

(4).该项目主要是基于注解，面向接口。

(5).配置文件的 dtd 是放于个人服务器，方便维护修改。（然而服务器并没有备案，dtd 也是超级简单，也是这学期刚学的 xml 设计运用上来了）

(6)该项目目前仅支持 mysql 数据库

# 2. 快速上手

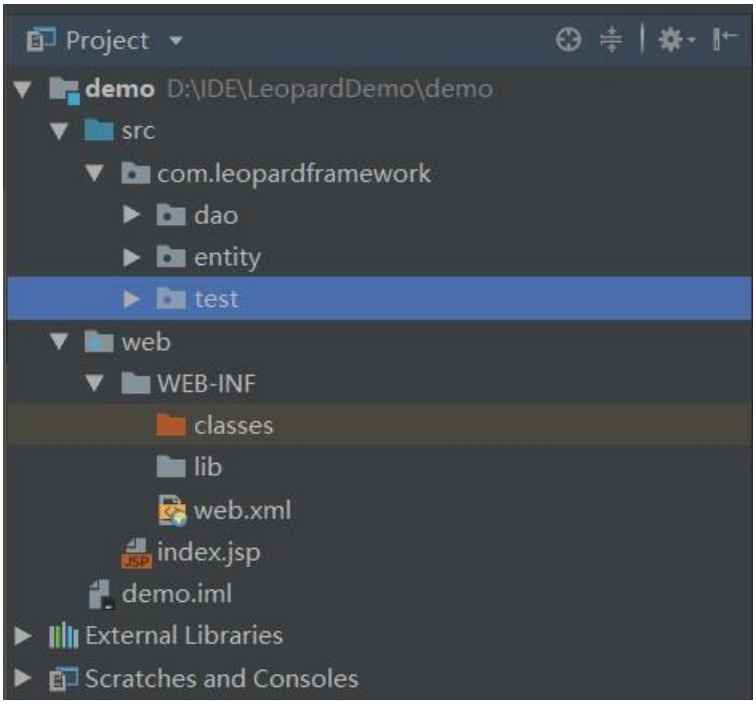
## 2.1Eclipse 下开发

1.没有安装 eclipse，下次再补上。

## 2.2IDEA 下开发

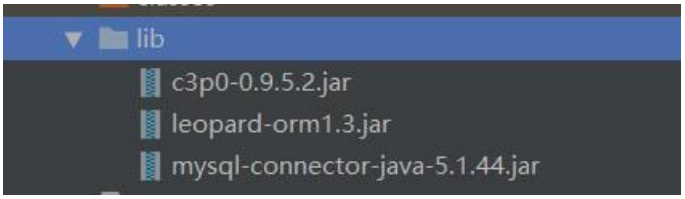
因为我们这个 leopard 是 jdbc 持久化层框架，下面我们创建一个简单的 web 项目来进行演示。

1. 创建 web 项目，结构图如下（怎么创建就不演示了）




2. 在 lib 文件夹中导入我们 leopard 的 jar 包和数据库连接驱动包。

（c3p0 连接池用到再导入）




3. 至此，我们就可以开始我们的项目了，把我们的 entity 对象类注解给配置上

```
@Table("user")  // @Table 一定要 value 对应数据库的表名 value为空则取类名
public class User {

    // @Column value 对应数据库的字段名 value为空则默认取变量名 isPrimary主键标识
    @Column(value = "id", isPrimary = Primary.AUTOINCREMENT)
    private Integer id;

    @Column("name")
    private String name;

    @Column
    private String address;
```

## 2.3Maven 开发（暂不支持）

# 3. 配置文件详解

## 3.1 配置文件

1.在我们的项目根路径下创建我们的配置文件（注：是我们的 java 文件根目录 ./src/）

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE leopard-config  PUBLIC "-//leopard.com//DTD Config 1.0//EN"
       "http://120.78.131.95/leopard/config/leopard.dtd">
<leopard-config>
    <!-- 数据源的配置-->
    <bean class="com.leopardframework.plugins.DBPlugin" id="dataSource">
        <property name="driver" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://127.0.0.1:3306/demo?characterEncoding=UTF-8"/>
        <property name="username" value="root"/>
        <property name="password" value="chg122345"/>
    </bean>
    <!-- c3p0数据源的配置-->
    <!--<bean class="com.leopardframework.plugins.c3p0.C3p0Plugin" id="dataSource">
        <property name="driver" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://127.0.0.1:3306/myshop?characterEncoding=UTF-8"/>
        <property name="username" value="root"/>
        <property name="password" value="chg122345"/>
        <property name="maxPoolSize" value="100"/>
        <property name="minPoolSize" value="20"/>
    </bean>-->
    <!--实体对象所在包-->
    <entity-package value="com.leopardframework.entity"/>
</leopard-config>
```

## 3.2 配置项

1.配置数据源，指定 id 为 dataSource 不能更改，要配置数据库的信息，propety 属性的 name 值固定为上图所示，就是 DBPlugin 的属性名。

2. bean 标签你也可以配置自定 bean 类，propety 属性的 name 的值就是 bean 的属性变量名， value 就是你要赋予的 bean 的属性变量的值。

3. entity-package 标签的 value 值配置你的实体类所在的包。

# 4. 逆向工程

## 4.1 逆向工程配置

1.逆向工程配置和上面配置基本一致

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE leopard-config  PUBLIC "-//leopard.com//DTD Config 1.0//EN"
       "http://120.78.131.95/leopard/config/leopard.dtd">
<leopard-config>
    <!-- 数据源的配置-->
    <bean class="com.leopardframework.plugins.DBPlugin" id="dataSource">
        <property name="driver" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://127.0.0.1:3306/demo?characterEncoding=UTF-8"/>
        <property name="username" value="root"/>
        <property name="password" value="chg122345"/>
    </bean>
    <!-- c3p0数据源的配置-->
    <!--<bean class="com.leopardframework.plugins.c3p0.C3p0Plugin" id="dataSource">
        <property name="driver" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://127.0.0.1:3306/myshop?characterEncoding=UTF-8"/>
        <property name="username" value="root"/>
        <property name="password" value="chg122345"/>
        <property name="maxPoolSize" value="100"/>
        <property name="minPoolSize" value="20"/>
    </bean>-->
    <generator>
        <target package="com.leopardframework.entity2" project="\demo\src\"></target>
    </generator>
</leopard-config>
```

2.数据源和上面一致，generator 里 target 标签的 package 配置你的工程生成的目标包，Project 属性配置你的工程目录所在位置。

## 4.2 工程生成

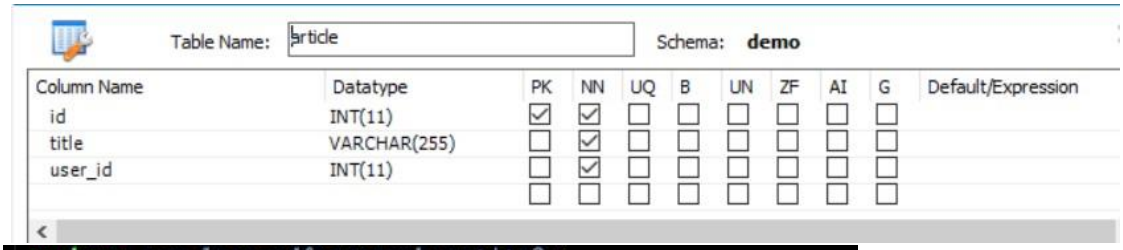
1.生成逆向工程这里需要引入文件的 io 包（commons-io-2.5.jar），具体如下：

```
public static void main(String[] args){
    GeneratorFactory factory=Factory.getGeneratorFactory([illegible]"classpath:leopard.xml"); //获取generator工厂
    try {
        factory.openGenerator();  //执行
    } catch (SQLException e) {
        e.printStackTrace();
    }


}
```

2.生成的 javabean 信息如下图（注：暂不会识别外键）：

（数据库表信息）：




```java
package com.leopardframework.entity2 ;

import com.leopardframework.core.annotation.*;
import com.leopardframework.core.enums.Primary;


/**
 *
 * @Copyright  (c) by Chen_9g (80588183@qq.com).
 * @Author  Leopard Generator
 * @DateTime  2018-04-20 22:00:12
 */
@Table("article")
public class Article {

    @Column(value="id",isPrimary = Primary.YSE)
    private Integer id;
    @Column("title")
    private String title;
    @Column("user_id")
    private Integer userId;

    public Article() {
    }
```

# 5. 封装的方法

## 5.1Insert 方法

```
SessionFactory factory=Factory.getSessionFactory("classpath:leopard.xml");// 获取Session工厂
private SqlSession  session=factory.openSession();  //开启新的session

public int saveUser(User user){
    try {
        int temp=session.Save(user);  //反回的数据库更新记录数
        session.Commit();   //对数据库的更新操作时记得提交session 默认开启了事物
        session.Stop();     //暂停session释放资源
        return temp;
    } catch (SqlSessionException e) {
        e.printStackTrace();
    }
    return 0;
}
```

```
public static void main(String[] args){
    User user=new User();
    user.setName("Leopard");
    user.setAddress("江西南昌");
    Dao dao=new Dao();
    int temp=dao.saveUser(user);  //调用UserDao的save方法 保存对象 User id我们设置是自增的，所以不用设user 的id值
    System.out.println("user:"+temp);  //打印记录  // ouput: user:1
    /*-----------------------------------------*/
    Article article=new Article();
    User a_user=new User();
    a_user.setId(1);   //把user的id值传入
    article.setId(10086); // article的id
    article.setTitle("Leopard测试");
    article.setUser(a_user); //外键把user传入 主要是user的id 存入数据库
    int temp2=dao.saveArticle(article);
    System.out.println("article: "+temp2); // ouput: article: 1

}
```

数据库的记录

| | id | name | address |
|---|---|---|---|
| ▶ | 1 | Leopard | 江西南昌 |
| | 2 | Leopard | 江西南昌 |

user表的记录

| | id | title | user_id |
|---|---|---|---|
| ▶ | 10086 | Leopard测试 | 1 |
| * | NULL | NULL | NULL |

article表的记录

## 5.2Delete 方法

```
public int delById(Integer id){
    try {
        int temp =session.Delete(User.class,id);//根据主键删除对象 批量删除可传入多个值
        session.Delete(User.class, ...objects: 1,2,3,4); //删除主键值为1 2 3 4 的记录
        session.Commit();
        session.Stop();
        return temp;
    } catch (SqlSessionException e) {
        e.printStackTrace();
    }
    return 0;
}
public int delUser(User user){
    try {
       int temp= session.Delete(user);  //根据对象信息进行删除 返回记录 （把条件都封装好）
        session.Commit();
        session.Stop();
        return temp;
    } catch (SqlSessionException e) {
        e.printStackTrace();
    }
    return 0;
}
```

## 5.3Update 方法

```
public int updateUserById(User user,Integer id){
    try {
        int temp= session.Update(user,id);  //根据主键对对象信息进行修改 返回记录（主键值默认不可修改,对象封装了新主键值也不会修改）
        session.Commit();
        session.Stop();
        return temp;
    } catch (SqlSessionException e) {
        e.printStackTrace();
    }
    return 0;
}
```

## 5.4Query 方法

```
public User getUserById(Integer id){
    try {
        User user=session.Get(User.class,id).get(0);  //根据主键查询 返回相应的对象 （主键可传多个，批量查询，返回list）
        session.Stop();
        return user;
    } catch (SqlSessionException e) {
        e.printStackTrace();
    }
    return null;
}

public User getUser(User u){
    try {
        User user=session.Get(u);  //根据对象信息查询 返回相应的对象 （把条件都封装好）
        session.Stop();
        return user;
    } catch (SqlSessionException e) {
        e.printStackTrace();
    }
    return null;
}

public List<User> getUserByWhere(){
    try {
        List<User> users=session.Get(User.class, "where id=? and name=?", 1,"Leopard");
        //自定义条件查询对象信息 返回相应的对象 （order by group 都可以用。后面参数按顺序传入）
        session.Stop();
        return users;
    } catch (SqlSessionException e) {
        e.printStackTrace();
    }
    return null;
}
```

```
public List<User> getAll(){
    List<User> list=new ArrayList<>();
    try {
        list =session.Get(User.class);  //查询所有的user 记录
        session.Stop();
    } catch (SqlSessionException e) {
        e.printStackTrace();
    }
    return list;
}
```

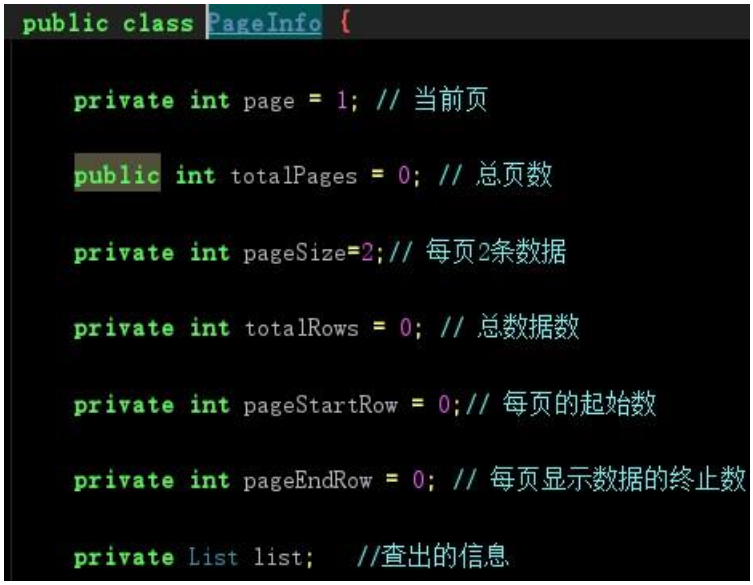
测试结果：

```
public static void main(String[] args){
    Dao dao=new Dao();
    User user=new User();
    System.out.println(dao.getUserById(1)); //输出 User{id=1, name='Leopard', address='江西南昌'}
     user.setId(2);
     user.setName("Leopard");
     System.out.println(dao.getUser(user)); //输出 User{id=2, name='Leopard', address='江西南昌'}
     List<User> users=dao.getAll();
     System.out.println(users);
     // 输出list [User{id=1, name='Leopard', address='江西南昌'}, User{id=2, name='Leopard', address='江西南昌'},
     // User{id=3, name='Leopard', address='江西南昌'}, User{id=4, name='Leopard', address='江西南昌'}]

}
```

# 6. 分页查询详解

## 6.1 分页信息类 pageInfo 介绍



```
public class PageInfo {

    private int page = 1; // 当前页

    public int totalPages = 0; // 总页数

    private int pageSize=2;// 每页2条数据

    private int totalRows = 0; // 总数据数

    private int pageStartRow = 0;// 每页的起始数

    private int pageEndRow = 0; // 每页显示数据的终止数

    private List list;    //查出的信息
```

```
public PageInfo(int totalRows, int pageSize) {
    init(totalRows, pageSize);// 通过对象记录总数划分
}

/** *//**
 * 初始化list，并告之该list每页的记录数
 * @param totalRows
 * @param pageSize
 */
public void init(int totalRows, int pageSize) {
    this.pageSize = pageSize;
    this.totalRows = totalRows;
    if ((totalRows % pageSize) == 0) {
        totalPages = totalRows / pageSize;
    } else {
        totalPages = totalRows / pageSize + 1;
    }

    if (totalRows < pageSize) {
        this.pageStartRow = 0;
        this.pageEndRow = totalRows;
    } else {
        this.pageStartRow =pageSize*(this.getPage()-1);
        this.pageEndRow = (this.getPage()*pageSize);
    }
}
```

## 6.2 分页方法使用

```
public PageInfo getUserToPage(int page,int pageSize){
    PageInfo pageInfo=null;
    try {
        pageInfo=session.Get(User.class,page,pageSize); //参数传入要查询的页码 每页显示的数量 指定用pageInfo接收
        session.Stop();
    } catch (SqlSessionException e) {
        e.printStackTrace();
    }
    return pageInfo;
}
```

测试结果：

```java
public static void main(String[] args){
    Dao dao=new Dao();
    PageInfo pageInfo=dao.getUserToPage(User.class, [illegible] 1, [illegible] 2); //查询第一页 每页显示2条记录
    System.out.println(pageInfo.getList());
   //输出 [User{id=1, name='Leopard', address='江西南昌'}, User{id=2, name='Leopard', address='江西南昌'}]

}
```

```java
public static void main(String[] args){
    Dao dao=new Dao();
    PageInfo pageInfo=dao.getUserToPage(User.class, [illegible] 2, [illegible] 3); //查询第2页 每页显示3条记录
    System.out.println(pageInfo.getList());
   //输出 [User{id=4, name='Leopard', address='江西南昌'}] 因为我们数据库总共才4条数据 所以查询第2页的时候只有一条记录

}
```

# 7. 多表连接(外键关联)

## 7.1 javabean 规则

```java
@Table
public class Article {

    @Column(isPrimary = Primary.YSB)
    private Integer id;

    @Column
    private String title;

    //外键要配置 relation relation值应与value 值保持一致
    //把外键对象设为成员变量 （m 2 one）多对一
    @Column(value = "user_id",relation = "user_id")
    private User user;
```

## 7.2 用法

1.其他操作都一样，查询操作时对外键的赋值是只赋其主键的值。

2.分页查询用法

```java
public PageInfo getArticleToPage(Class<?>cls1,Class<?>cls2,int page,int pageSize){
    PageInfo pageInfo=null;
    try {
        //第一个类即为我们要查的类，第二个类是外键对应的类
        pageInfo=session.Get(cls1,cls2,page,pageSize); //参数传入要查询的页码 每页显示的数量 指定用pageInfo接收
        session.Stop();
    } catch (SqlSessionException e) {
        e.printStackTrace();
    }
    return pageInfo;
}
```

测试结果

```
public static void main(String[] args){
    Dao dao=new Dao();
    PageInfo pageInfo=dao.getArticleToPage(Article.class,User.class, [illegible] 1, [illegible] 3); //查询第1页 每页显示3条记录
    System.out.println(pageInfo.getList());
    //输出 [Article{id=10086, title='Leopard测试', user=User{id=1, name='Leopard', address='江西南昌'}}]
    // 因为我们数据库总共只有1条数据 所以查询的时候只有一条记录

}
```

| | id | title | user_id |
|---|---|---|---|
| | 110 | hello 110 | 3 |
| | 10000 | Hello 电信 | 1 |
| | 10010 | Hello 联通 | 2 |
| ▶ | 10086 | Leopard测试 | 1 |

article表中的数据

```
public static void main(String[] args){
    Dao dao=new Dao();
    PageInfo pageInfo=dao.getArticleToPage(Article.class,User.class, [illegible] 1, [illegible] 4); //查询第1页 每页显示4条记录
    System.out.println(pageInfo.getList());
    //输出 [Article{id=10000, title='Hello 电信', user=User{id=1, name='Leopard', address='江西南昌'}},
    // Article{id=10086, title='Leopard测试', user=User{id=1, name='Leopard', address='江西南昌'}},
    // Article{id=10010, title='Hello 联通', user=User{id=2, name='Leopard', address='江西南昌'}},
    // Article{id=110, title='hello 110', user=User{id=3, name='Leopard', address='江西南昌'}}]
}
```

查出的结果是按照 user 的 id 索引，应为我们的 user id 是设置自增的，被标识为索引条件。

# 8. 扩展

在我们的 com.leopardframework.util 工具包中还写了大量经常使用的工具

类：


util
ArrayUtil
CastUtil
ClassUtil
CollectionUtil
DateUtil
FileUtil
MapUtil
MD5Util
ObjectUtil
PropsUtil
ReflectionUtil
StringUtil